LA-UR-94-3449

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: STATUS OF CFDLIB PERFORMANCE TESTS ON THE T3D

AUTHOR(S): N(ely) T. Padial, T-3
B(ryan) A. Kashiwa, T-3
D(ouglas) B. Kothe, T-3

SUBMITTED TO: *Cray User Group Conference, CEA/CEL-V (Commissariat a l'Energie Atomique, Centre de'Etudes de Limeil-Valenton, Tours, France, October 10–14, 1994*

Los Alamos

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO. 836 R4
ST. NO. 2629 5/81

STATUS OF CFDLIB PERFORMANCE TESTS ON THE T3D

Nely T. Padial, Bryan A. Kashiwa, and Douglas B. Kothe

Theoretical Division
Los Alamos National Laboratory †
Los Alamos, NM 87545

ABSTRACT

CFDLIB is a collection of two- and three- dimensional computer codes for problems in computational fluid dynamics. The library handles a wide range of flow regimes both single and multiphase, from fully incompressible to hypersonic, chemically reacting or inert materials. The method uses the Implicit Continuous-fluid Eulerian scheme (ICE) which becomes essentially identical to the Marker and Cell (MAC) method in the incompressible limit. The method, a Finite-Volume scheme, with cell-centered state variables, utilizes an Arbitrary-Lagrangian-Eulerian (ALE) split computational cycle in the sense that the mesh is allowed to move in an arbitrary fashion. In addition, CFDLIB employs a multiblock data structure that proved extremely convenient for implementation on parallel computers. CFDLIB is now available in sequential or parallel environments, is written in standard FORTRAN 77, and is highly portable. Results are given for a selection of specialized applications that illustrate the kind of problems of interest to CFDLIB users. Finally, initial results are given for performance studies comparing the Cray Y-MP, T3D, and an IBM workstation cluster. For certain problems, a Y-MP-equivalent real solution time is achieved with less than eight processors on the T3D.

I. Introduction.

The long-term goal of this work is to develop a practical simulation capability for reactive multiphase flow problems of day-to-day interest in US industry. The Los Alamos Hydrocode Library *CFDLIB* (Computational Fluid Dynamics LIBrary) is being developed in order to accomplish this goal. This paper reports the early status of comparative performance testing of the code library.

Classical examples of multiphase flow are bubbles of gas rising in a liquid, and solid grains falling in a liquid. A three-phase flow exists when the gas, solid, and liquid are all coflowing. Reactive multiphase flow occurs, for example, when the gas becomes capable of dissolving in the liquid, or when the solid grains act as a catalyst to reactions that take place among different chemical species in the liquid. Chemical Engineers make use of devices that rely on reactive multiphase flow processes for a great many purposes in industry. Typically a vessel called a reactor is used to contain the process, with reactants injected at one place, and products extracted at another. An important parameter in the design of these systems is the reactor efficiency, a measure of the effectiveness with which the reactant stream is converted into the product stream.

These reactive multiphase flow processes can be represented by solutions to a system of conservation equations for mass, momentum, energy, and other variables such as catalyst activity, that describe the averaged 'state' of the flow at a point. Adequate description of the state typically requires keeping track of the momentum and energy of each phase, as well as the mass concentration of numerous species making up each material. Hence there might be a minimum of 31 variables necessary to define the averaged state at a point, for a three-phase problem in which there are three species making up each phase. The numerical means by which time-dependent solutions are obtained requires that the reactor volume be subdivided into a collection of small volumes, called cells, in which the conservation equations are to be satisfied. Hence, to define the state at a point, one needs 31 words of memory per cell in the foregoing case. Sufficient resolution of the flow dynamics may easily require on the order of 10^7 such cells. Furthermore, the numerical methodology often may require another 10^2 or so words per cell, so that the memory needed rapidly approaches 10^{10} words for problems of this sort.

In addition to a single solution to the state, we seek a sequence of solutions separated by small time increments, adding considerable complexity to the problem. This time sequence, which may be 10^5 steps long, serves to produce a simulation of the process that is invaluable in understanding the nature of the process, improving the efficiency or creating a whole new process.

Hence the problem of creating the process simulation is highly computationally intensive, not only from the memory standpoint but also in terms of the time required to reproduce the many hundreds of arithmetic operations that advance the state from one point in the time sequence to the next.

The hydrocode library *CFDLIB* is being developed at Los Alamos for these computationally intensive problems. The next section is a brief description of the physical models contained in the library, the numerical method employed, and the data structure used in *CFDLIB*. Section III contains a description the steps taken to convert the original sequential version to one capable of running on a selection of distributed–memory platforms that include UNIX workstation clusters, and the Cray T3D. Section IV describes three typical applications in reactive multiphase flow. Performance comparisons for two of these problems are given in Sec. V. In the final section we briefly summarize this work and present a few conclusions.

II. Equations, Method, Data Structure.

The equations used to describe the reactive multiphase flow problems shown here are derived from a statistical average of the exact conservation laws (Kashiwa and Rauenzahn, 1994). The discontinuous microscopic nature of the instantaneous fields is smoothed by the statistical averaging process. The result is a set of conservation equations for interpenetrating continua, each phase representing one of the continua which are interacting with each other through exchange terms. One of the significant challenges on the theoretical side of this work is in the modeling of those exchange terms, in addition to closure modeling for other terms that represent effects of fluid turbulence. This is an area that remains somewhat open, and one which may be significantly improved by a simulation capability of the sort described here.

CFDLIB employs a solution method based on a class of schemes called finite–volume (FV) methods. In an FV scheme, the integral form of the conservation equations is solved on a domain overlaid by a mesh of small 'cells' each of which is treated as a control volume. The flows of material, momentum, and energy are computed in a strongly conservative fashion since the flux leaving one cell is exactly that which enters its neighbor across a common cell boundary. Because of this strong conservation, FV schemes are among those most commonly used in CFD. A detailed description of the numerical recipe used in this work has been given elsewhere (Kashiwa, Padial, Rauenzahn and VanderHeyden, 1994).

An important aspect of the efficient solution of the discrete equation is in the manner by which the data is stored in the computer, called the data structure, since the rate at which data can be accessed and used depends on where it resides in the computer memory. A multiblock data format has been selected for use in *CFDLIB*. In this, the physical domain is partitioned into 'blocks' (that can be envisioned as a brick) of regular structured mesh having an arbitrary logical size (length, width, and height). Each block is connected to its neighbor by way of 'ghost cells' that are invisible, and in which is stored a duplicate copy of the state, corresponding to the neighboring block's data. Within each block, the data is stored in contiguous arrays so that access to neighboring data is accomplished with pre-determined offsets, making indirect addressing unnecessary. This data arrangement is ideal for vector processing since a sweep over each block is performed with a single array index, allowing long vector lengths to be used. Details of this procedure have been published elsewhere (Addessio, *et al.*, 1991).

Besides enabling the calculation on rather complex geometric domains, and apart from enabling efficient vector processing, this data structure lends itself naturally to distributed memory processing. Each block, or group of blocks, can be directed for processing on a separate processing element, and a message passing protocol used to ship data into neighboring ghost cells. With this in mind, the inter–block communication routines in *CFDLIB* are designed around a message–passing template. First, a sending buffer array is loaded with the host block's state data; second, that array is copied into a receiving buffer array allocated for the receiving block; and third, the receiving buffer is unloaded into the receiving block's ghost cell storage space. On a single–memory computer the copy is performed directly; on a distributed memory computer the copy is performed by whatever networking means may be available. Hence, porting among various parallel computers is limited to changing the way in which the copy is performed.

A description of the porting process, the various data transfer schemes available, and performance are the subject of the next sections.

III. Parallelization Strategies.

Apportioning of the work among the available processors forms the first step for the parallelization of the code. For more than one processor, the first one (PE 0) has the task of input/output as well as the calculation of quantities needed by the other processors. In the division of labor, the other processors do the calculation of at least one block. If the total number of blocks is NBLKS and the number of processors available for this part of the work is NUPRO, each processor PE, if greater than zero, takes care of the blocks varying from PE to NBLKS in increments of NUPRO.

Most of *CFDLIB* routines deal with calculations in each block. These routines need not to be changed. Some of the routines are drivers that send work to the various processors and have to be

changed very little. They must recognize the fact that different blocks may be calculated on different processors. Some of the routines are communication routines that receive information from and send information to neighboring blocks. A layer of ghost cells surrounds the block boundaries. Real cell data values of one block are transferred to the ghost cells of an adjacent block. *CFDLIB* requires the sharing of three types of information: vertex–centered arrays, cell–centered arrays, and face–centered arrays. In this way, different communication routines deal with each different case.

The Parallel Virtual Machine (PVM) software system (Beguelin *et al.*, 1991) provides an attractive vehicle for utilizing a collection of heterogeneous computer systems concurrently and also for operating massively parallel machines. However, PVM libraries on different machines require vendor specific interfaces making porting between machines or cluster cumbersome. This difficulty could be avoided by making use of UNCOL (UNiversal Communication Library) developed by Meltz and Korias (these proceedings). This is a library of message passing primitives that uses PVM routines arranged in such a way as to allow codes to run unchanged on a variety of machines. UNCOL consists of a main program that initializes the tasks involved in the computation before calling the user program as a subroutine which, in turn, calls the routines provided in the library for communication. Although UNCOL provides many functions, CFDLIB uses only the point-to-point send and receive messages and the broadcast from one task to every other one. UNCOL is especially appealing for keeping high portability without sacrificing excessively the efficiency of the calculation compared to the direct use of the PVM routines.

When there is communication between two blocks, each block must give and receive information. Using the message passing system, a processor loads a variable that needs to be communicated in a temporary array and sends this array through one of the UNCOL routines (SND_INT_MSG or SND_REAL_MSG). The processor with the adjacent block receives this variable through another of the UNCOL routines (RCV_INT_MSG or RCV_REAL_MSG) in another temporary array which is, then, loaded in its final destination.

However, when using the Cray T3D, the ability of addressing the memory globally offers the opportunity of improving efficiency considerably. A paper by R. W. Numrich (1994a) on the Cray T3D address-space offers valuable suggestions for using the feature. Numrich wrote, at our request, a synchronization routine for an set of processors p1 to p2. The method uses the language F^{--} (Numrich, 1994b).

The applications presented in this paper use eight modules of *CFDLIB*. These modules involve one hundred and twenty six routines. One hundred and three of these routines are exactly the same in both the sequential and parallel versions. Only six of the modified routines require very extensive modifications.

IV. Description of Test Problems.

Our initial performance studies consist of three multiphase and one single–phase flow problems: a two–dimensional gas–liquid separator simulation, a three-dimensional bubble column startup, a three–dimensional gas–liquid-solid riser simulation in a fluid catalytic cracking (FCC) unit, and a single–phase Sedov blast wave. Each of these exhibits different forms of complexity in the computation as described next.

Separator. The gas–liquid separator, a device commonly used in chemical processing as a means of extracting gas bubbles from a liquid, consists of a tank having a central downcomer pipe, as shown in Fig. 1. A bubbly flow stream is introduced at the bottom, and a disengagement surface forms near the top, above which a port for gas removal is placed. The goal is to let the gas bubbles migrate to the surface before the liquid exits via the central downcomer pipe. Efficiency is measured by the amount of gas removed before the flow reaches the downcomer and leaves the tank.

The first test problem computes the separator flow for a period of time, on an interior domain subdivided into 28 blocks of mesh. This allows us to place a different number of blocks on a processor, in order to observe a measure of scaling effectiveness on the parallel processor. Figure 2 shows typical flow field.

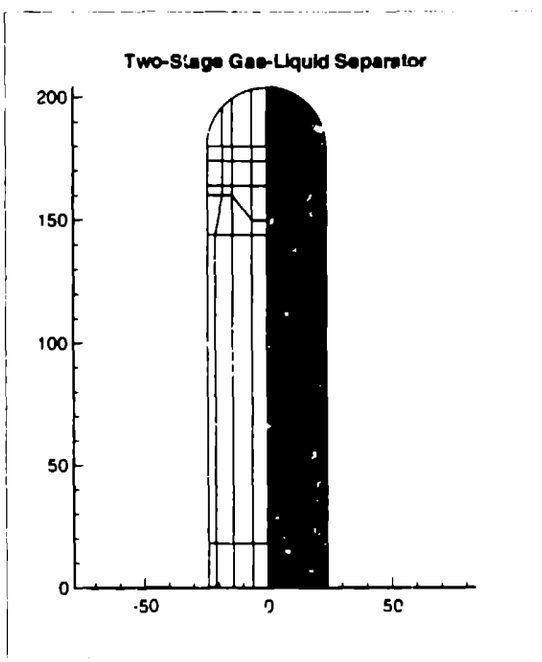

Figure 1. Twenty–eight block, axisymmetric mesh for the separator problem. The heavy lines indicate block boundaries.

3D Bubble Column The second test problem centers on the startup of a bubble column in 3D. A bubble column, another common device in chemical processing, consists of a tank of liquid that has a stream of gas bubbles introduced across the bottom. Figure 3 shows the boundaries of the five–block mesh used in the calculation. The purpose here is to use the rapid upward motion of the bubbles to stir the liquid. A chemical reaction between the gas and the liquid may also be of interest, but is not computed in this test case. A time sequence of the flow startup shown in Fig. 4 demonstrates the complexity of the flow patterns that developed. Initially the pool of liquid is at rest, and the gas flow is applied uniformly across the bottom. Very early on the gas rises as a coherent collection of bubbles in a one-dimensional fashion. As time increases this one-dimensionality is perturbed by numerical errors; this triggers the Rayleigh–Taylor instability that rapidly

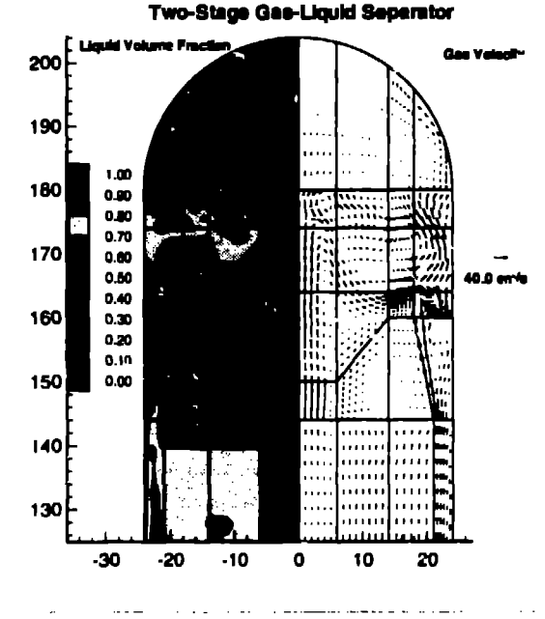

Figure 2. Typical flow field, at a particular instant, in the separator simulation problem. The topmost part of the mesh is shown.

breaks the one-dimensional motion into a complex, swirling, three–dimensional motion. The flow naturally organizes itself so that a bubble–rich section exists near the center of the column, where the gas flow is strongly upward and drags the liquid upward with it. At the sides of the column resides a liquid–rich layer where the the liquid falls until it reaches the bottom where it is again lifted by the gas, and so on.

Since the bubble column domain is subdivided into five blocks, each of which contains a number of cells very much larger than those in the separator case, the area to volume ratio is smaller, in terms of the amount of data to be communicated, than in the first case. Hence this gives a measure of performance where one has taken care to minimize communication and maximize processor–wise calculation.

FCC Injection Region. In this 3D problem, an upward flow of gas carries solid grains of hot catalyst into the bottom of a vertical riser pipe. The cylindrical computational domain is partitioned into five mesh blocks as in the bubble column simulation. Liquid oil, 400°F cooler than the hot catalyst, is injected at four locations approximately one pipe diameter up from the entrance of the pipe. Injection occurs at an angle of 60 degrees relative to the vertical and 45 degrees relative to the pipe wall, which induces an upward swirling motion. The oil is injected just below its vaporization temperature at twenty times the velocity of the upward-moving gas-solid mixture. The hot catalyst rapidly heats the oil, causing it to vaporize. This process generates an enormous volume of oil vapor that rushes upward, accelerating the slower-moving catalyst grains.

Figures 5 and 6 display volume fraction isosurfaces for the FCC injection simulation. The injected oil is not allowed to vaporize in the case shown in Fig. 5, in contrast with the case of Fig. 6, where vaporization occurs rapidly, generating large pockets of gas-rich regions.

This is a three-phase (liquid-solid-gas) problem that is highly time-unsteady. Since the cracking process occurs in the vapor as the mixture accelerates up the riser pipe, these types of simulations address crucial issues for the chemical engineer such as the liquid oil penetration and mixing, oil vapor generation rate, and vapor distribution.

Blast Wave. The final test problem is designed to give some better indication of the scaling behavior of *CFDLIB* on parallel computers. For this we compute the dynamics of a blast wave originating from a point at which energy is deposited in an otherwise uniform, stationary, medium. This is a standard test problem for hydrocodes, named after L. I. Sedov (1959), who first gave the analytic solution in the case of a perfect gas. We conduct the calculation on a domain consisting of a ten by ten array of blocks, each of which is made up of a ten by ten array of cells. Figure 5 shows the mesh and some aspects of the solution at a certain time, shortly after deposition of the energy.

V. Performance Comparison.

A performance measure of particular interest is the real time necessary to obtain solutions. A convenient way of expressing this is the grind time, defined as the real (wall–clock) time per simulation time step, per mesh cell. (For a single dedicated processor, we assume the real time and cpu time are equivalent.) Each of the foregoing test problems was run on a single processor of the Cray Y-MP, running UNICOS 7.C.3. For purposes of contrast here, we let the single–processor Y-MP grind time t_{ymp} serve as the normalization factor. That is, we define the nondimensional time t as the multiprocessor grind time t_{mp}

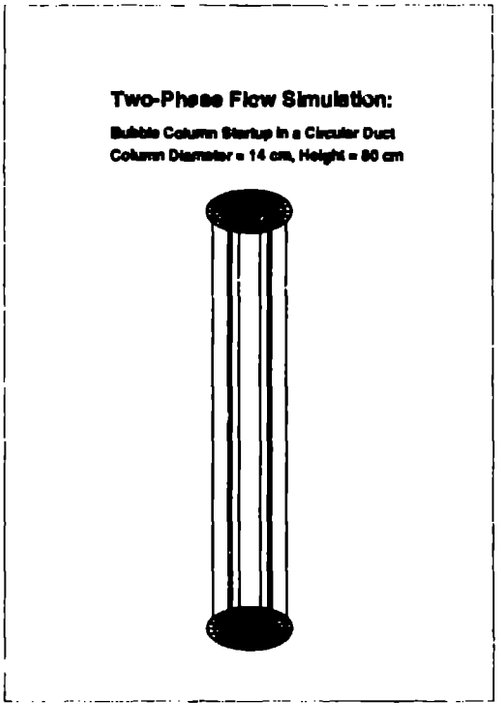

Figure 3. Mesh configuration for the 3D bubble column startup. The top and bottom planes are shown, with vertical lines tracing the corners of block boundaries.

divided by t_{ymp}; hence a unit value of $t = t_{mp}/t_{ymp}$ constitutes breakeven. Smaller values of t indicate an overall time savings.

We display a performance comparison for the separator problem, and for the Sedov problem. These represent two very different kinds of problem: one is communication–bound, and the other is not. The multiphase flow in the separator is an incompressible flow, requiring the solution of an implicit (mesh–wide) problem for the pressure field, at each step in time. The Sedov problem is an explicit compressible flow case not requiring such a solution for the pressure. In addition, the mesh for the separator is far from optimum in terms of the surface to volume ratio of cells in each block, whereas the Sedov mesh is perfectly balanced with a much smaller number of surface cells compared to the number of cells in each block. The Y-MP grind time for the implicit separator problem is 285 μs per cycle per cell and that for the explicit Sedov problem is 45 μs. The difference arises mainly from the many (a few hundred) data communications performed every time step in the implicit case, versus the single data communication each step of the explicit case. Since a large number of problems in industry require implicit solutions, and a large number of problems in defense require explicit solutions, we report on both cases here.

Figure 7 we plot $\log_{10} t$ versus $\log_2 p$, where p is the number of processors, for the 28 block separator problem. Recall that we dedicate one of those processors to the meager tasks of I/O and control functions, so for $p = 2$ all of the work is done on one processor. We note that for this case none of the systems tested acnieves breakeven ($\log_{10} t = 0$) since the large surface to volume ratio typical of the small blocks of mesh tends to cause a communication bottleneck that is further aggravated by the implicit problem.

Figure 8 is a plot of $\log_{10} t$ versus $\log_2 p$ for the Sedov problem. In this instance the Cray T3D achieves a clear breakeven (Y-MP time–equivalence) with less than eight processors and almost a factor of ten speed up ($\log_{10} t \rightarrow -1$) with 64 processors.

VI. Summary.

We presented a brief introduction to the hydrocode library CFDLIB and early performance results of computations on the Cray T3D with both PVM and F^{--} communication protocols and on an IBM workstation cluster with PVM. We used as a baseline the performance of the single–processor Cray–YMP. We observe a linear speedup in the case of an explicit problem, for which inter–processor data communication is performed once at each time step of the simulation. However, we do not achieve a YMP breakeven for an implicit problem, where many inter–processor data communications are required each time step and for which a poorly suited mesh is used. We expect that a three–dimensional implicit problem, as the 3D FCC problem having a small surface to volume ratio for cell, would achieve breakeven for a number of processors between eight and sixteen.

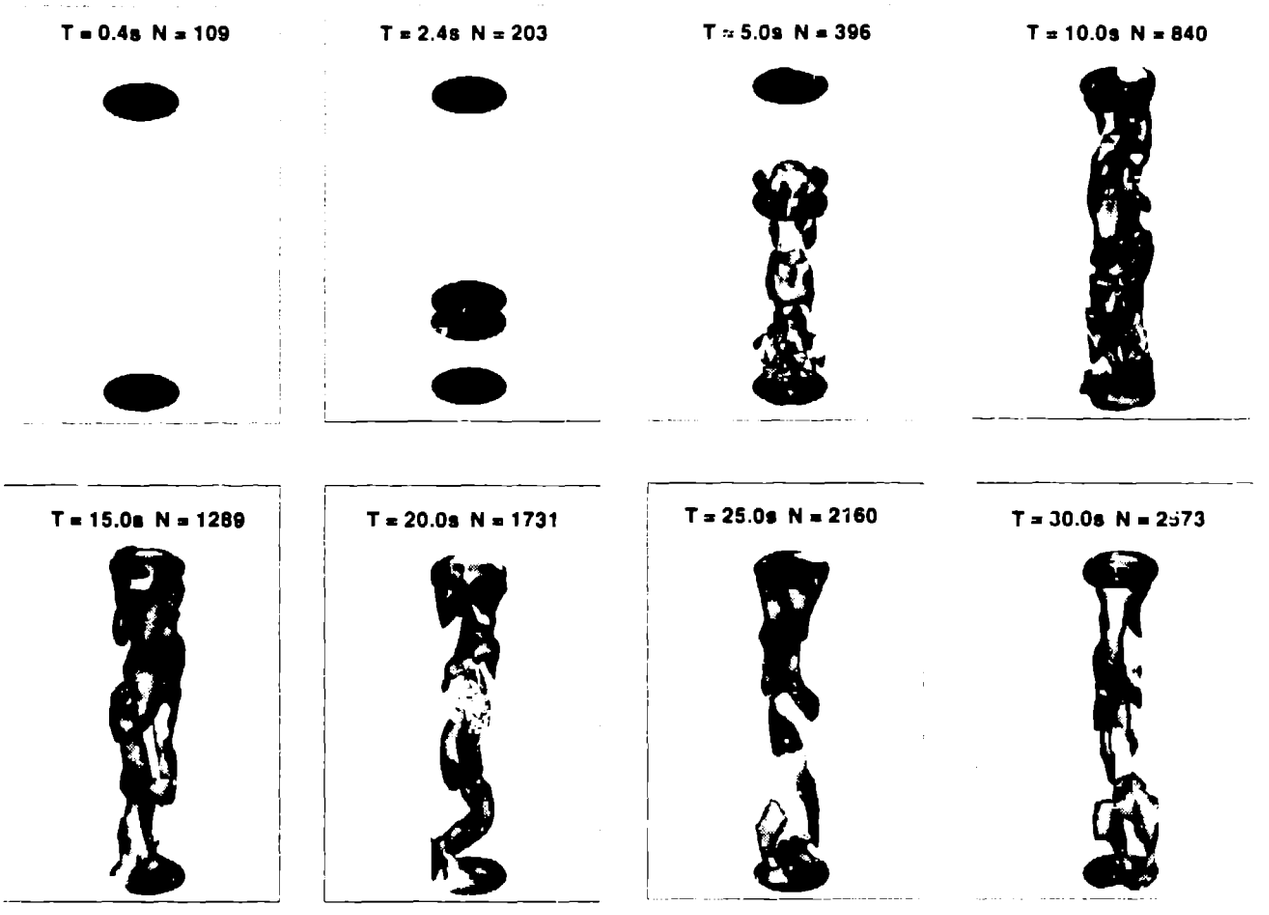

Figure 4. Time sequence of results from the bubble column simulation. Isosurfaces of 25% gas volume fraction are shown.

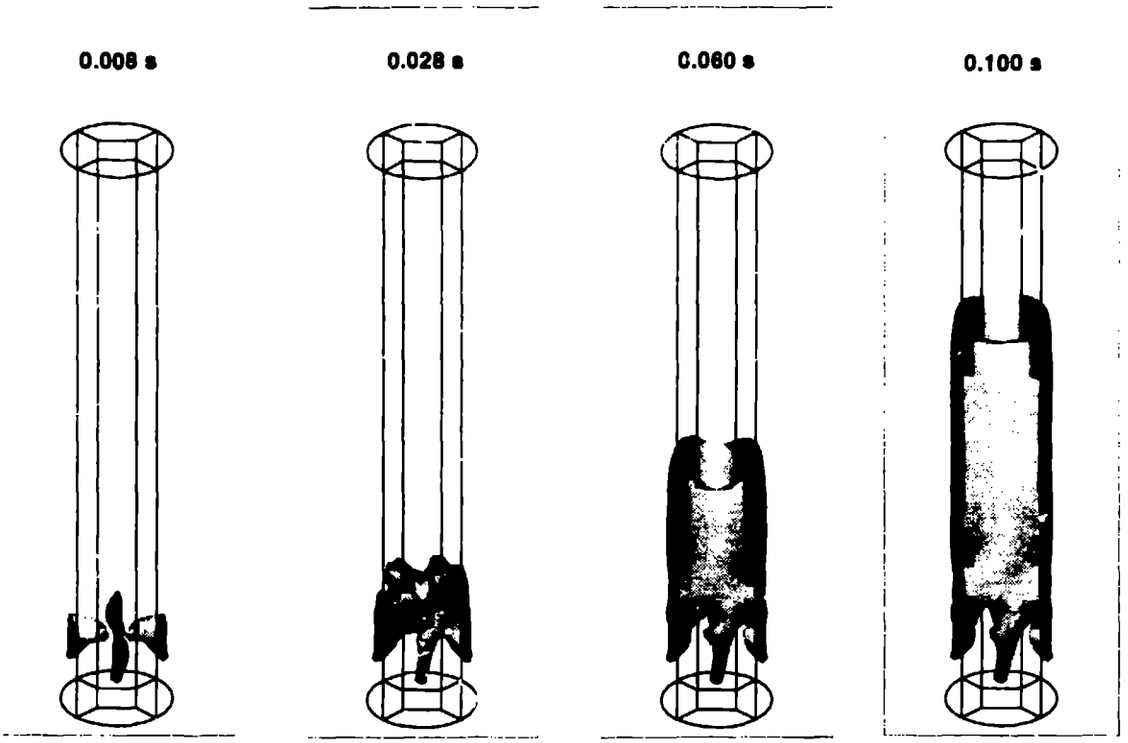

Figure 5. Time sequence of liquid volume fraction isosurfaces (15%) near the injection region of an FCC riser. Liquid is injected from four locations at high speeds relative to an upwarding-moving hot mixture of gas and solid catalyst. For illustration here, liquid vaporization is suppressed in this example. Lines denote mesh block boundaries.

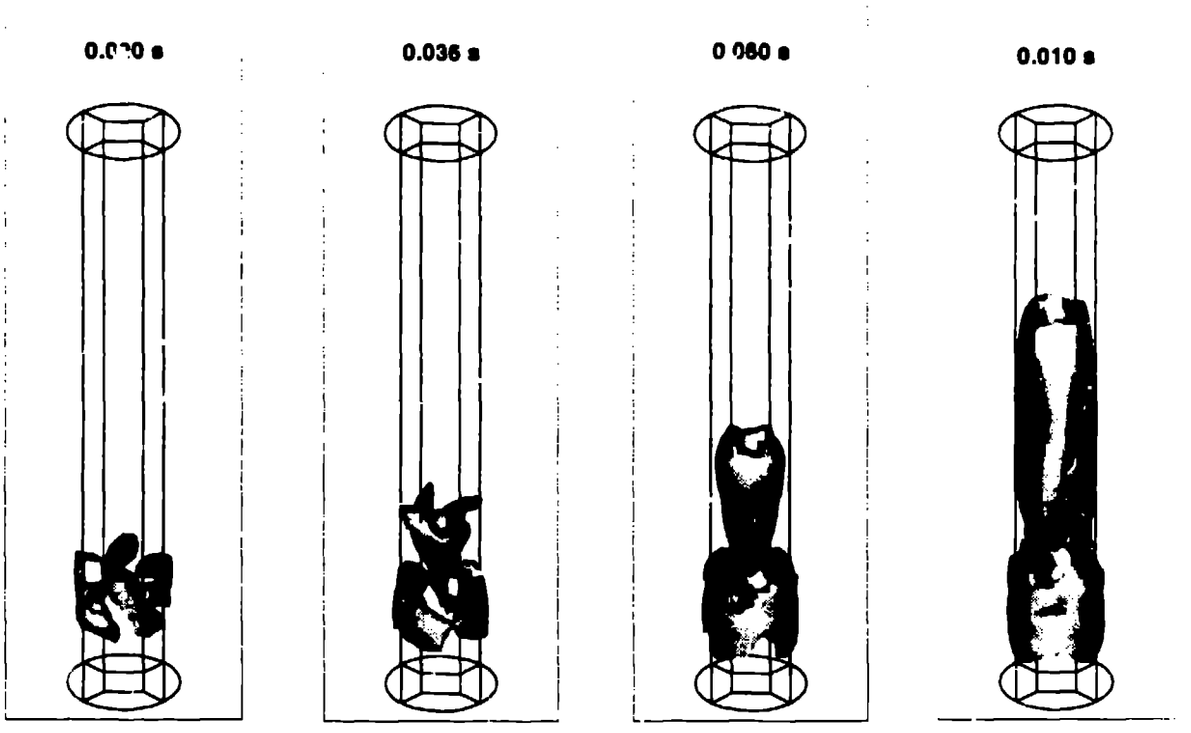

Figure 6. Time sequence of gas volume fraction isosurfaces (65%) near the injection region of an FCC riser. In contrast to the simulation of Fig. 5, the injected liquid is allowed to vaporize, and as a result liquid flashes violently, producing pockets of gas–rich regions.

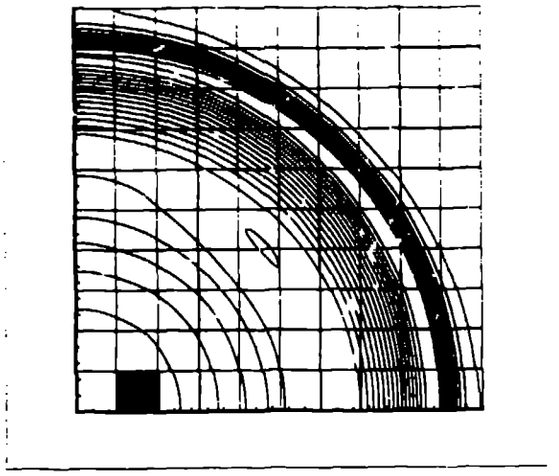

Figure 7. Mesh, blocks, and contours of constant pressure for the Sedov blast wave problem.

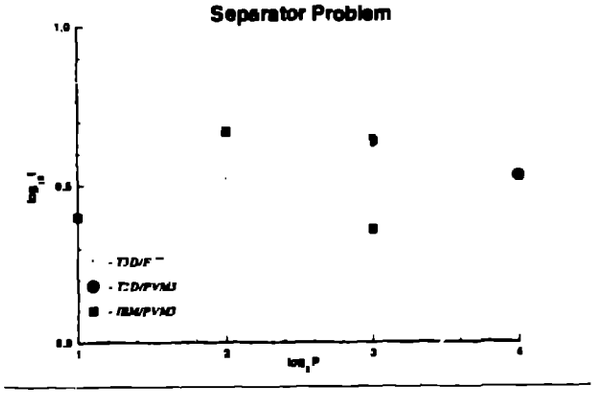

Figure 8. Performance comparison for the separator problem.

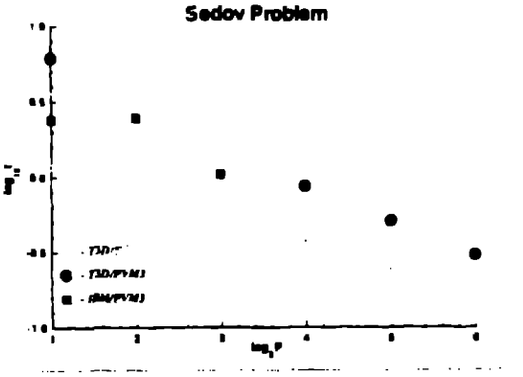

Figure 9. Performance comparison for the Sedov problem.

ACKNOWLEDGMENTS

Support for this work has come in part from a number of sources, notably the U.S. Departments of Energy, and Defense, The Amoco Oil Company, and Molten Metal Technology, Inc. There are numerous colleagues to thank for helpful discussions over a period of many years. Notable among them are John Baumgardner who created the innovative data structure that has proved so useful in this work; Bertrand Meltz (and colleagues) who had the insight and persistence to created the UNCOL library which makes portability among parallel computers a possibility; and Brian VanderHeyden for creating the input for the separator problem. Thanks go to Geert Weenes and colleagues at Cray Research Inc., for their support in using the new T3D. We also thank our many users of the Los Alamos Hydrocode Library *CFDLIB* for the interesting variety of problems that have provided the testbed for the results shown here.

REFERENCES

Addessio, F. L., J. R. Baumgardner, J. K. Dukowicz, N. L. Johnson, B. A. Kashiwa, R. M. Rauenzahn, and C. Zemach, *CAVEAT: A Computer Code for Problems with Large Distortion and Internal Slip*, Los Alamos National Laboratory Report LA-10613-MS, Rev. 1, 1992.

Beguelin, Adam, Jack Dongarra, Al Geist, Robert Manchek, and Vaidy Sunderam, *A User's Guide to PVM Parallel Virtual Machine*, ORNL/TM-11826, Oak Ridge National Laboratory, Oak Ridge, TN, 1991.

Kashiwa, B. A., and R. M. Rauenzahn, "A Multimaterial Formalism", in C. T. Crowe, ed., *Numerical Methods in Multiphase Flows 1994*, The American Society of Mechanical Engineers, New York, 1994.

Kashiwa, B. A., N. T. Padial, R. M. Rauenzahn, and W. B. VanderHeyden, "A Cell-Centered Method for Multiphase Flow Simulations", in C. T. Crowe, ed., *Numerical Methods in Multiphase Flows 1994*, The American Society of Mechanical Engineers, New York, 1994.

Meltz, Bertrand, and Samuel Kortas, These Proceedings, 1994.

Numrich, R. W., *The Cray T3D Address Space and How to Use It*, Technical Report, Cray Research, Inc., 1994a.

Numrich, R. W., *F**: A Parallel Fortran Language*, Technical Report, Cray Research, Inc., 1994b.

Sedov, L. I., *Similarity and Dimensional Methods in Mechanics*, Academic Press, New York, 1959.